Terraform Task

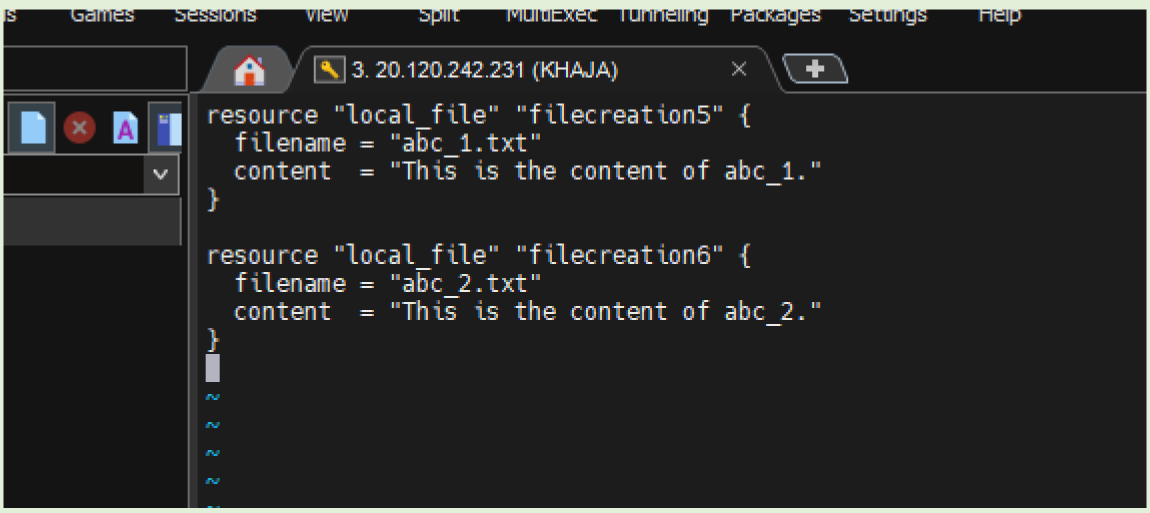
Name:Shaik Khaja Basha
Batch : Batch 11
Date : 19.07.2025
Task : File creations in .tf & make corrections

1. **Create a file and Subsequent apply, destroy**

Ans: for subsequent apply we ill create a file name file_kbs.tf

Enter the content in the file_kbs.tf like

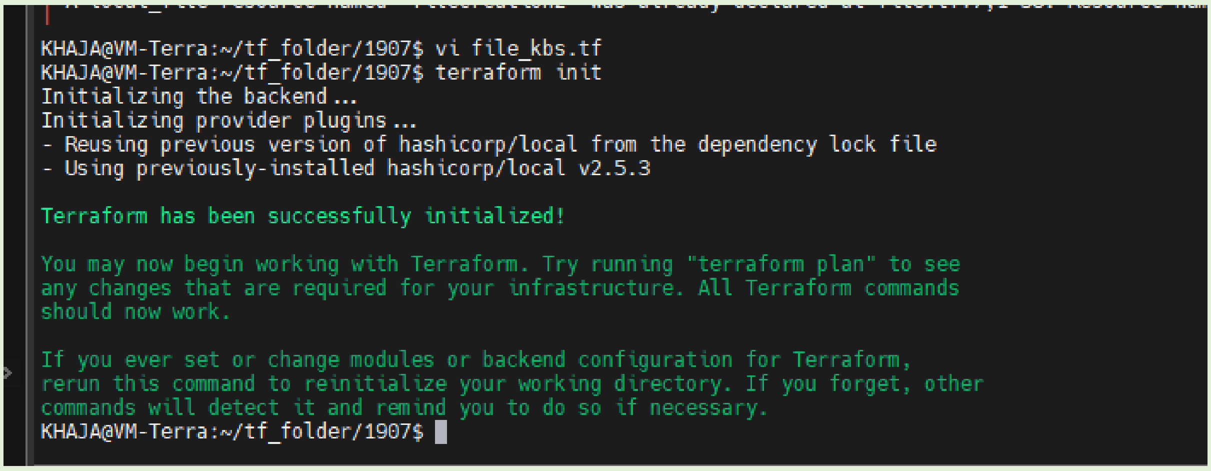
```
resource "local_file" "filecreation1" {
  filename = "abc_1.txt"
  content  = "This is the content of abc_1."
}


resource "local_file" "filecreation2" {
  filename = "abc_2.txt"
  content  = "This is the content of abc_2."
}
```



```
resource "local_file" "filecreation5" {
  filename = "abc_1.txt"
  content  = "This is the content of abc_1."
}

resource "local_file" "filecreation6" {
  filename = "abc_2.txt"
  content  = "This is the content of abc_2."
}

~
~
~
~
```

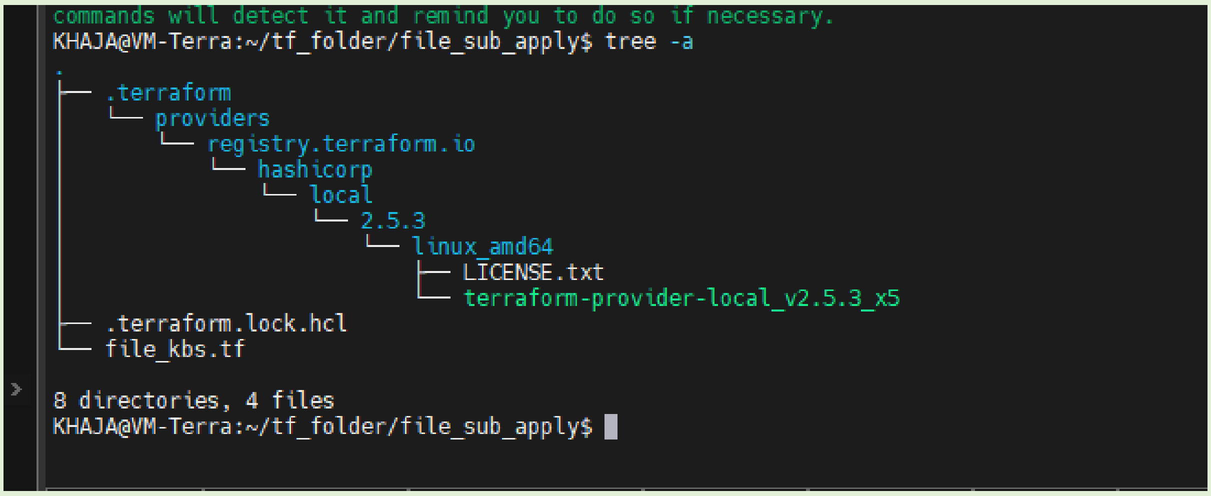
Initialize Terraform using command

- terraform init



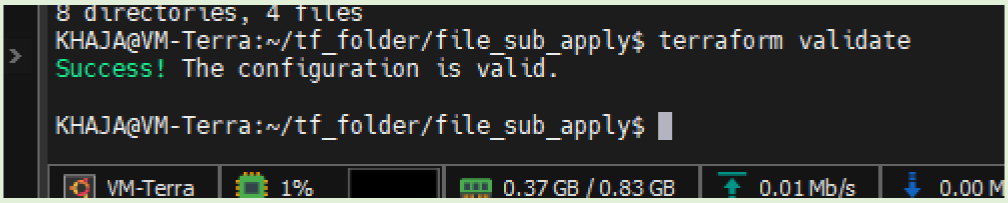
```
KHAJA@VM-Terra:~/tf_folder/1907$ vi file_kbs.tf
KHAJA@VM-Terra:~/tf_folder/1907$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Using previously-installed hashicorp/local v2.5.3

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/1907$
```

Check the contents using tree -a



```
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ tree -a
.
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               └── local
│                   └── 2.5.3
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-local_v2.5.3_x5
├── .terraform.lock.hcl
└── file_kbs.tf

8 directories, 4 files
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

Run the command terraform validate

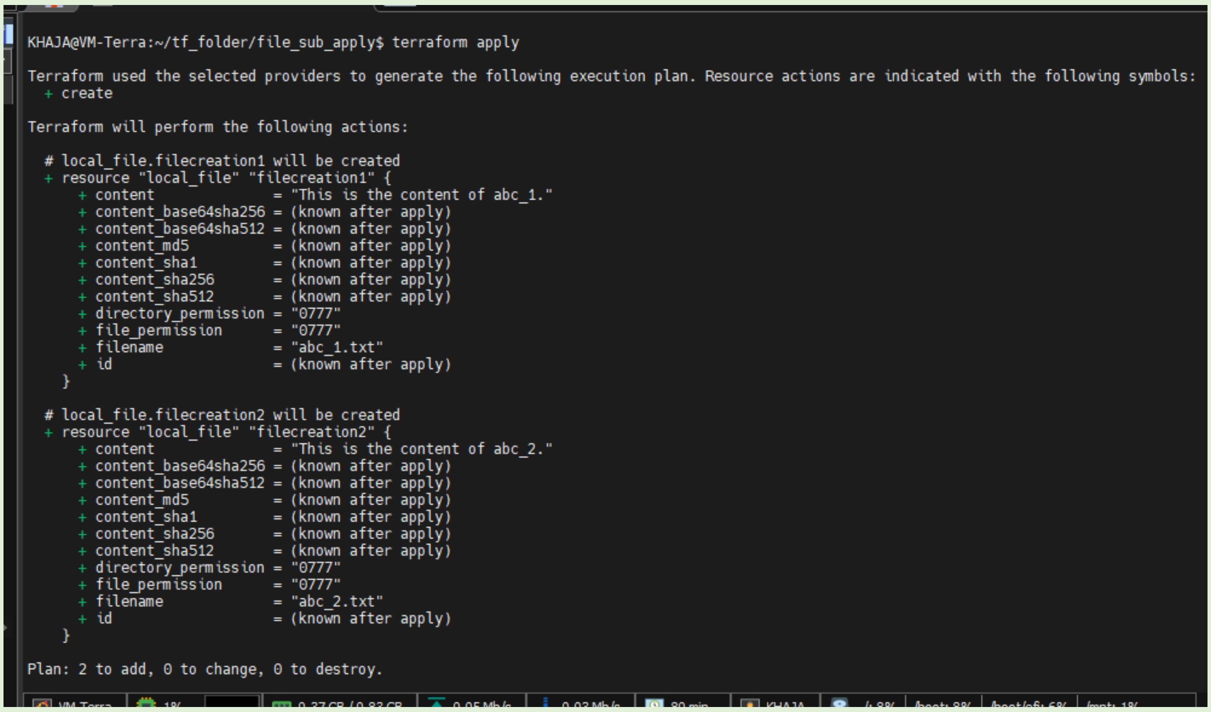


```
8 directories, 4 files
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform validate
Success! The configuration is valid.

KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

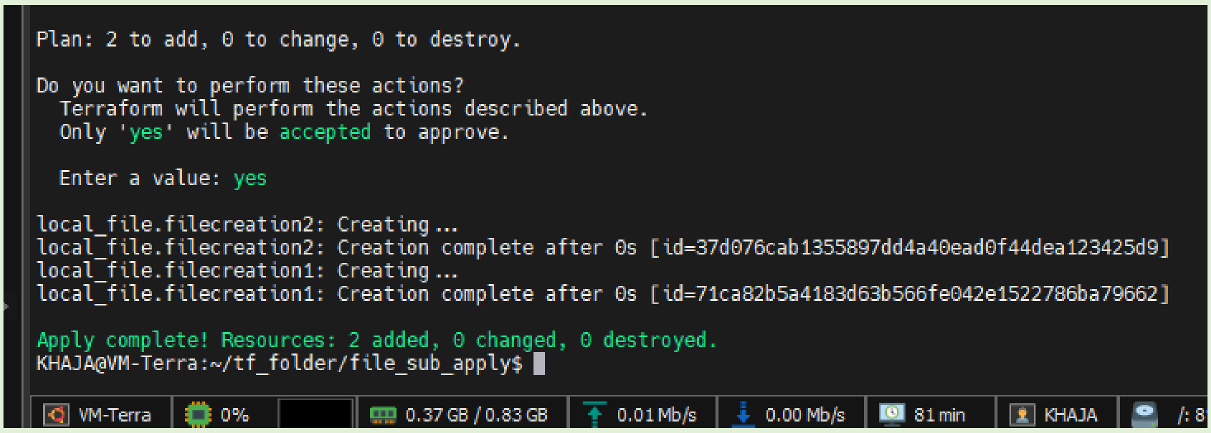
terraform plan

- it will show a preview or dry run.
- It shows like What Terraform wants to create, change, or delete.
- No changes are made to our real environment when running plan.

Run the command terraform plan



```
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # local_file.filecreation1 will be created
  + resource "local_file" "filecreation1" {
      + content              = "This is the content of abc_1."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc_1.txt"
      + id                   = (known after apply)
    }

  # local_file.filecreation2 will be created
  + resource "local_file" "filecreation2" {
      + content              = "This is the content of abc_2."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc_2.txt"
      + id                   = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

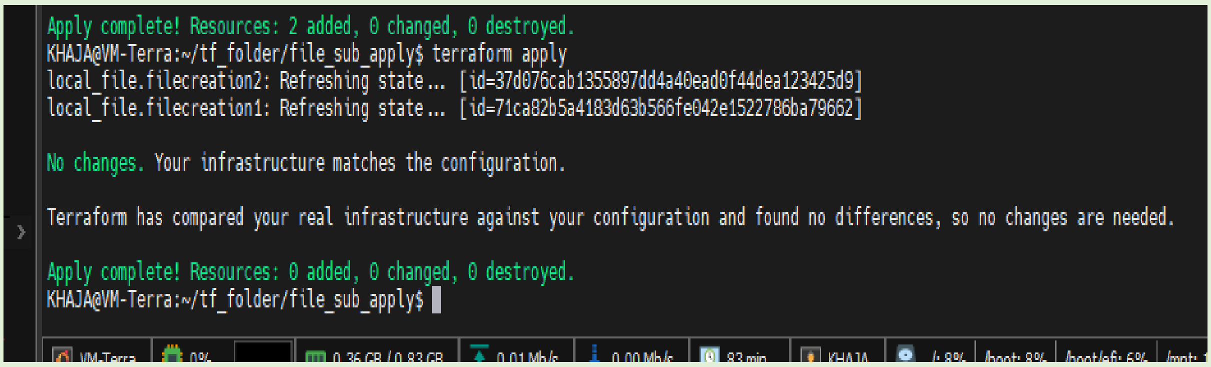


```
Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.filecreation2: Creating...
local_file.filecreation2: Creation complete after 0s [id=37d076cab1355897dd4a40ead0f44dea123425d9]
local_file.filecreation1: Creating...
local_file.filecreation1: Creation complete after 0s [id=71ca82b5a4183d63b566fe042e1522786ba79662]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

VM-Terra | 0% | 0.37 GB / 0.83 GB | 0.01 Mb/s | 0.00 Mb/s | 81 min | KHAJA | /: 8

Subsequent apply

- It will show that No changes are made to our real environment when running plan.



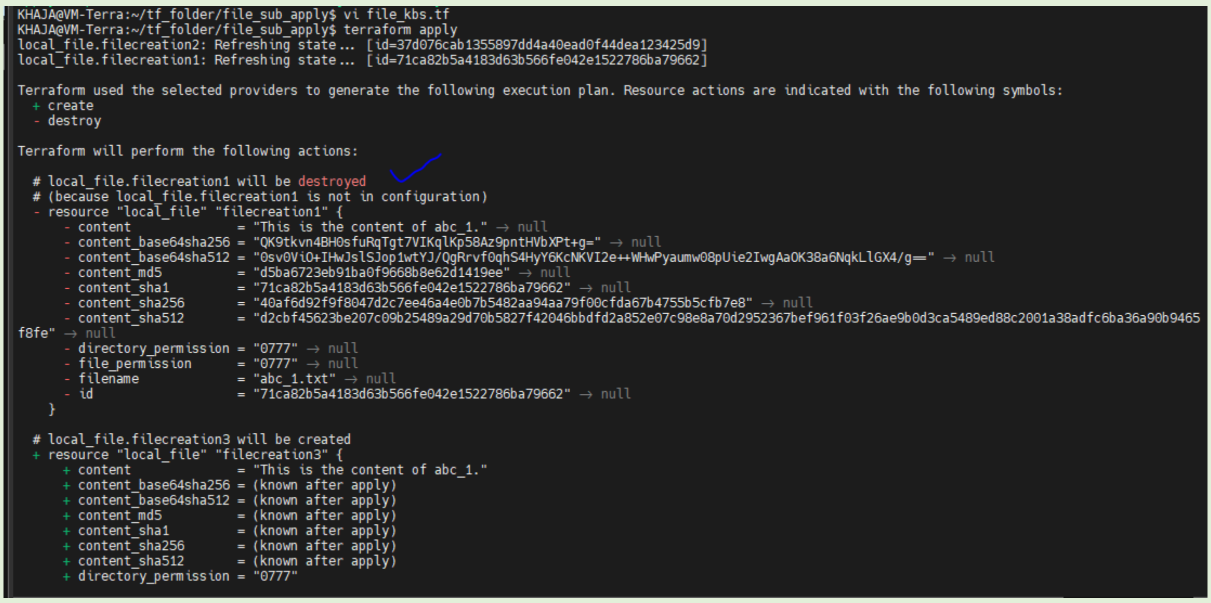
```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform apply
local_file.filecreation2: Refreshing state... [id=37d076cab1355897dd4a40ead0f44dea123425d9]
local_file.filecreation1: Refreshing state... [id=71ca82b5a4183d63b566fe042e1522786ba79662]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

Terraform tracks resource definitions using **resource blocks**, not by file content or filename. So:

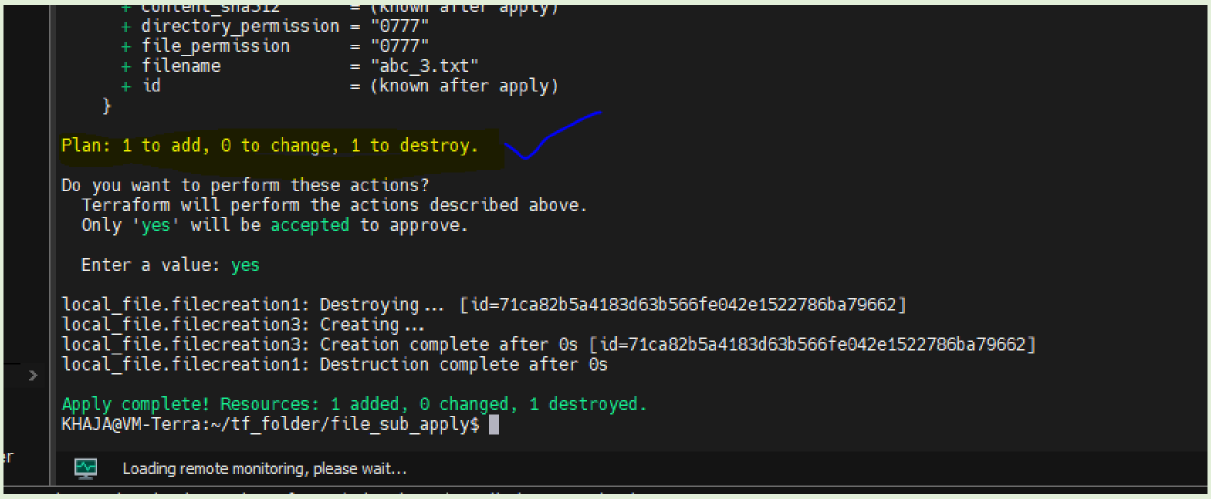
- **Removing a block** = Terraform **destroys** that resource
- **Adding a new block** = Terraform **creates** that resource
- Even if the **content is same**, if the **resource name changes**, Terraform treats it as new.

Now, in the updated file_kbs.tf, you removed filecreation1 and added:

```
resource "local_file" "filecreation3" {
  filename = "abc_3.txt"
  content  = "This is the content of abc_1."
}
```



```
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ vi file_kbs.tf
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform apply
local_file.filecreation2: Refreshing state... [id=37d076cab1355897dd4a40ead0f44dea123425d9]
local_file.filecreation1: Refreshing state... [id=71ca82b5a4183d63b566fe042e1522786ba79662]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
  - destroy

Terraform will perform the following actions:

  # local_file.filecreation1 will be destroyed
  # (because local_file.filecreation1 is not in configuration)
  - resource "local_file" "filecreation1" {
      - content              = "This is the content of abc_1." → null
      - content_base64sha256 = "QK9tkvn4BH0sfuRqTgt7VIKqlKp58Az9pntHVbXPt+g=" → null
      - content_base64sha512 = "0sv0ViO+IHwJslSJop1wtYJ/QgRrvf0qhS4HyY6KcNKVI2e++WHwPyaumw08pUie2IwgAaOK38a6NqkLlGX4/g==" → null
      - content_md5          = "d5ba6723eb91ba0f9668b8e62d1419ee" → null
      - content_sha1         = "71ca82b5a4183d63b566fe042e1522786ba79662" → null
      - content_sha256       = "40af6d92f9f8047d2c7ee46a4e0b7b5482aa94aa79f00cfda67b4755b5cfb7e8" → null
      - content_sha512       = "d2cbf45623be207c09b25489a29d70b5827f42046bbdfd2a852e07c98e8a70d2952367bef961f03f26ae9b0d3ca5489ed88c2001a38adfc6ba36a90b9465
f8fe" → null
      - directory_permission = "0777" → null
      - file_permission      = "0777" → null
      - filename             = "abc_1.txt" → null
      - id                   = "71ca82b5a4183d63b566fe042e1522786ba79662" → null
    }

  # local_file.filecreation3 will be created
  + resource "local_file" "filecreation3" {
      + content              = "This is the content of abc_1."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
```

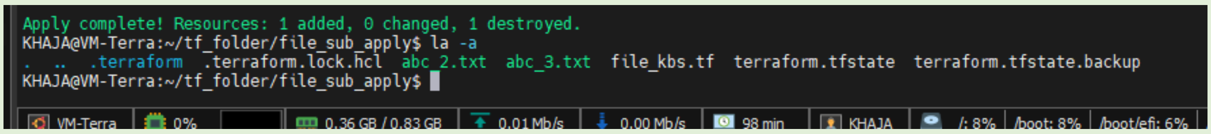


```
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc_3.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.filecreation1: Destroying... [id=71ca82b5a4183d63b566fe042e1522786ba79662]
local_file.filecreation3: Creating...
local_file.filecreation3: Creation complete after 0s [id=71ca82b5a4183d63b566fe042e1522786ba79662]
local_file.filecreation1: Destruction complete after 0s

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

Loading remote monitoring, please wait...

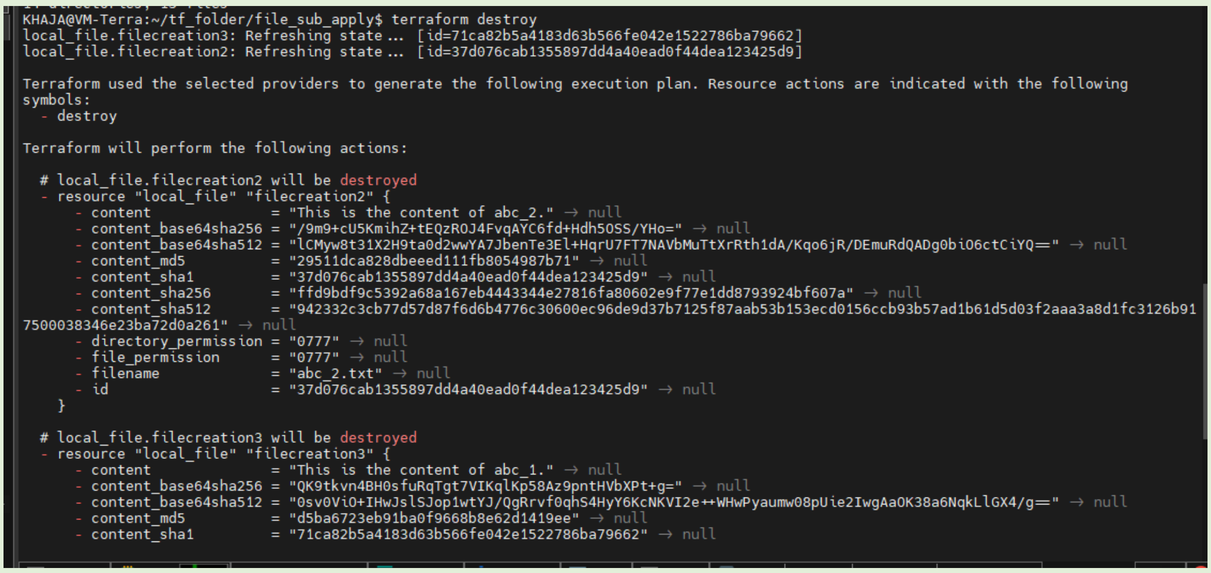
Output

As a result when we change and apply , it will remove the change file and create a new resource

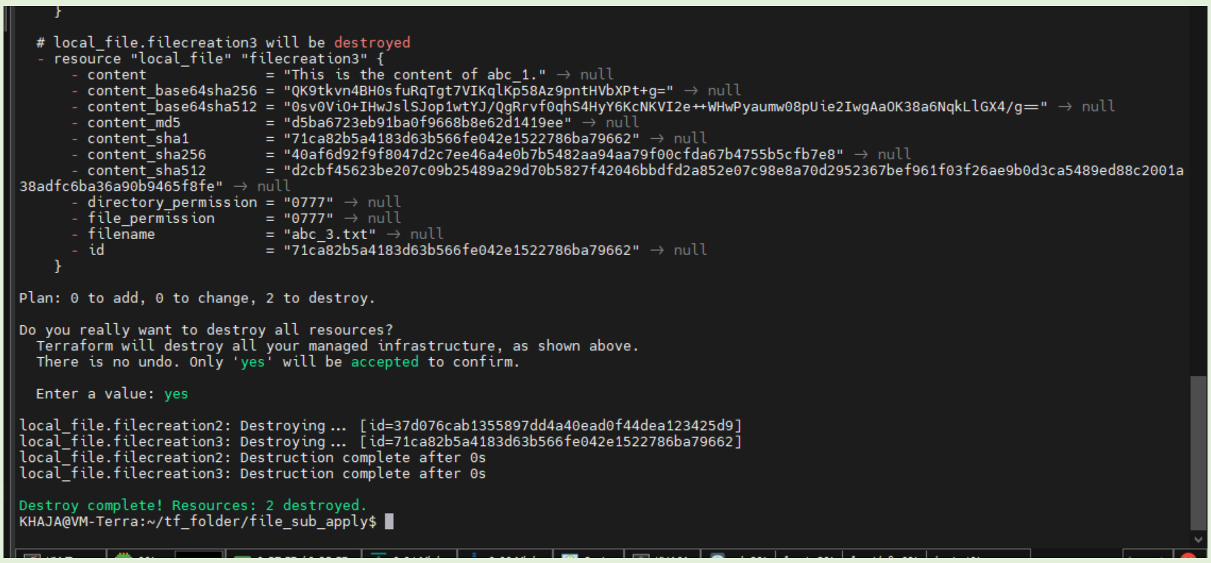


```
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ la -a
.  ..  .terraform  .terraform.lock.hcl  abc_2.txt  abc_3.txt  file_kbs.tf  terraform.tfstate  terraform.tfstate.backup
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

Use terraform detroy to destroy the resources

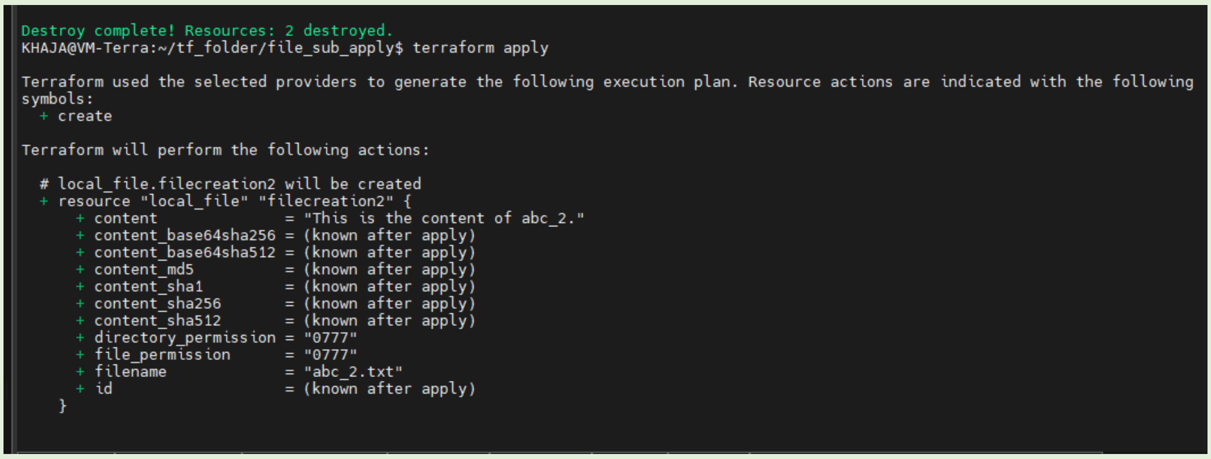


```
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform destroy
local_file.filecreation3: Refreshing state... [id=71ca82b5a4183d63b566fe042e1522786ba79662]
local_file.filecreation2: Refreshing state... [id=37d076cab1355897dd4a40ead0f44dea123425d9]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  - destroy

Terraform will perform the following actions:

  # local_file.filecreation2 will be destroyed
  - resource "local_file" "filecreation2" {
      - content              = "This is the content of abc_2." -> null
      - content_base64sha256 = "/9m9+cU5KmihZ+tEQzROJ4FvqAYC6fd+Hdh5OSS/YHo=" -> null
      - content_base64sha512 = "lCMyw8t31X2H9ta0d2wwYA7JbenTe3El+HqrU7FT7NAVbMuTtXrRth1dA/Kqo6jR/DEmuRdQADg0bi06ctCiYQ==" -> null
      - content_md5          = "29511dca828dbeeed111fb8054987b71" -> null
      - content_sha1         = "37d076cab1355897dd4a40ead0f44dea123425d9" -> null
      - content_sha256       = "ffd9bdf9c5392a68a167eb4443344e27816fa80602e9f77e1dd8793924bf607a" -> null
      - content_sha512       = "942332c3cb77d57d87f6d6b4776c30600ec96de9d37b7125f87aab53b153ecd0156ccb93b57ad1b61d5d03f2aaa3a8d1fc3126b91
7500038346e23ba72d0a261" -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = "abc_2.txt" -> null
      - id                   = "37d076cab1355897dd4a40ead0f44dea123425d9" -> null
    }

  # local_file.filecreation3 will be destroyed
  - resource "local_file" "filecreation3" {
      - content              = "This is the content of abc_1." -> null
      - content_base64sha256 = "QK9tkvn4BH0sfuRqTgt7VIKqlKp58Az9pntHVbXPt+g=" -> null
      - content_base64sha512 = "0sv0ViO+IHwJslSJop1wtYJ/QgRrvf0qhS4HyY6KcNKVI2e++WHwPyaumw08pUie2IwgAaOK38a6NqkLlGX4/g==" -> null
      - content_md5          = "d5ba6723eb91ba0f9668b8e62d1419ee" -> null
      - content_sha1         = "71ca82b5a4183d63b566fe042e1522786ba79662" -> null
```

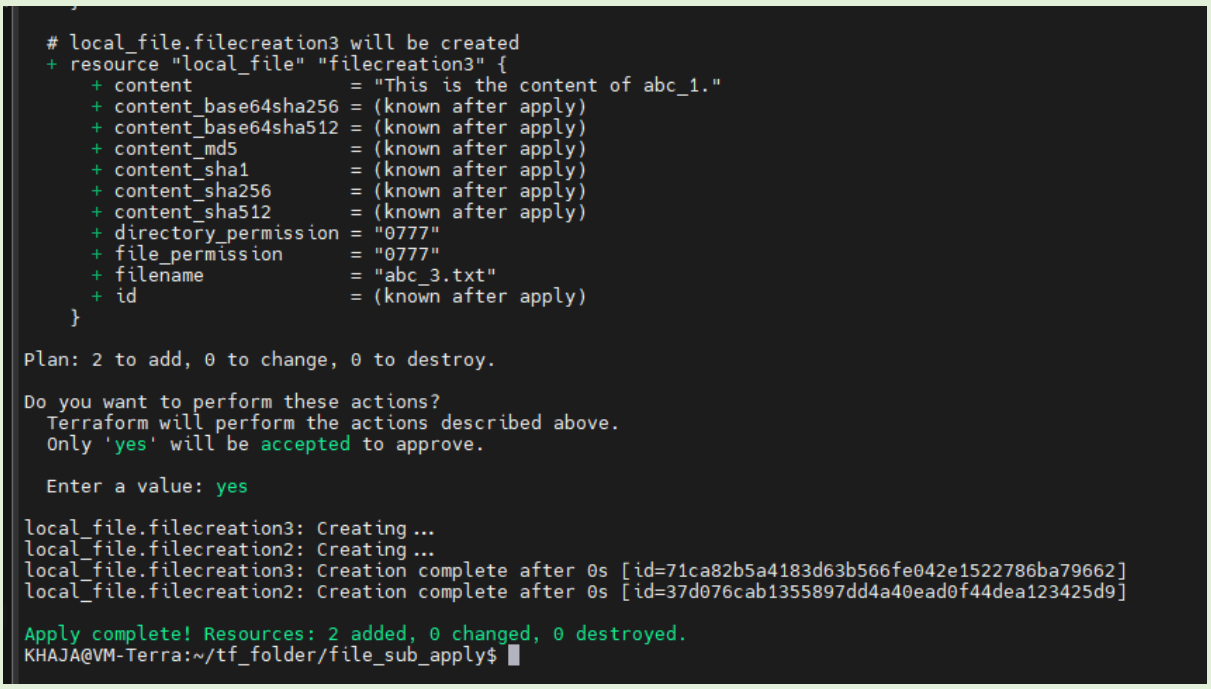


```
    }

  # local_file.filecreation3 will be destroyed
  - resource "local_file" "filecreation3" {
      - content              = "This is the content of abc_1." -> null
      - content_base64sha256 = "QK9tkvn4BH0sfuRqTgt7VIKqlKp58Az9pntHVbXPt+g=" -> null
      - content_base64sha512 = "0sv0ViO+IHwJslSJop1wtYJ/QgRrvf0qhS4HyY6KcNKVI2e++WHwPyaumw08pUie2IwgAaOK38a6NqkLlGX4/g==" -> null
      - content_md5          = "d5ba6723eb91ba0f9668b8e62d1419ee" -> null
      - content_sha1         = "71ca82b5a4183d63b566fe042e1522786ba79662" -> null
      - content_sha256       = "40af6d92f9f8047d2c7ee46a4e0b7b5482aa94aa79f00cfda67b4755b5cfb7e8" -> null
      - content_sha512       = "d2cbf45623be207c09b25489a29d70b5827f42046bbdfd2a852e07c98e8a70d2952367bef961f03f26ae9b0d3ca5489ed88c2001a
38adfc6ba36a90b9465f8fe" -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = "abc_3.txt" -> null
      - id                   = "71ca82b5a4183d63b566fe042e1522786ba79662" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

local_file.filecreation2: Destroying... [id=37d076cab1355897dd4a40ead0f44dea123425d9]
local_file.filecreation3: Destroying... [id=71ca82b5a4183d63b566fe042e1522786ba79662]
local_file.filecreation2: Destruction complete after 0s
local_file.filecreation3: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

If we run **terraform apply** again with the same configuration the file will be recreated



```
Destroy complete! Resources: 2 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.filecreation2 will be created
  + resource "local_file" "filecreation2" {
      + content              = "This is the content of abc_2."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc_2.txt"
      + id                   = (known after apply)
    }
```



```
  # local_file.filecreation3 will be created
  + resource "local_file" "filecreation3" {
      + content              = "This is the content of abc_1."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc_3.txt"
      + id                   = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.filecreation3: Creating...
local_file.filecreation2: Creating...
local_file.filecreation3: Creation complete after 0s [id=71ca82b5a4183d63b566fe042e1522786ba79662]
local_file.filecreation2: Creation complete after 0s [id=37d076cab1355897dd4a40ead0f44dea123425d9]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ 
```

2. explore these providers -> local, random, null

Ans: Definition:

Terraform providers are essentially plugins that allow Terraform to interact with various infrastructure platforms (like cloud providers, SaaS providers, or APIs).

- **Local Providers:**

  Definition:

Local providers handle tasks within the same environment as Terraform, often for testing or local resource management.

- **Usage:**

They can also be used for other local operations, such as generating random passwords, IDs, or other unique values.

- Generating configuration files (e.g., for apps, scripts, or cloud-init).
- Saving outputs (e.g., certificates, keys) to disk.

Key Resource: local_file

```
resource "local_file" "config" {
  filename = "app.conf"
  content  = "key=value"  # Dynamic content via variables/templates
  file_permission = "0644"  # Linux file permissions (optional)
}
```

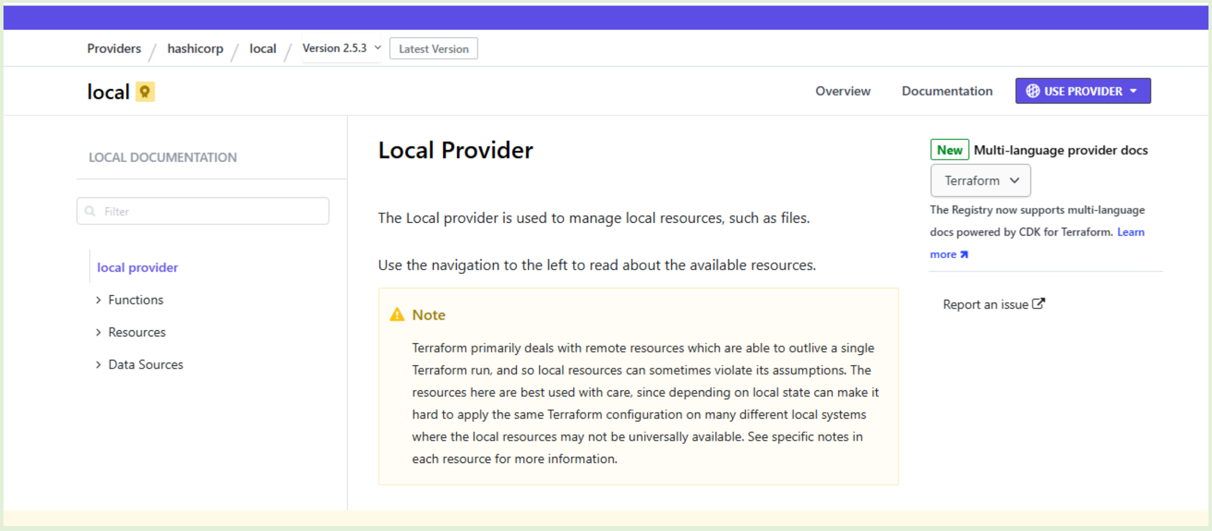
Lifecycle Control:

```
lifecycle {
  ignore_changes = [content]  # Prevents overwrites if file is modified externally
}
```

Example: Generate a Dynamic Script

```
resource "local_file" "startup_script" {
  filename = "start.sh"
  content = <<-EOT
    #!/bin/bash
    echo "Hello, ${var.username}!"
  EOT
}
```

They are the certified trusted providers



Local Provider

The Local provider is used to manage local resources, such as files.

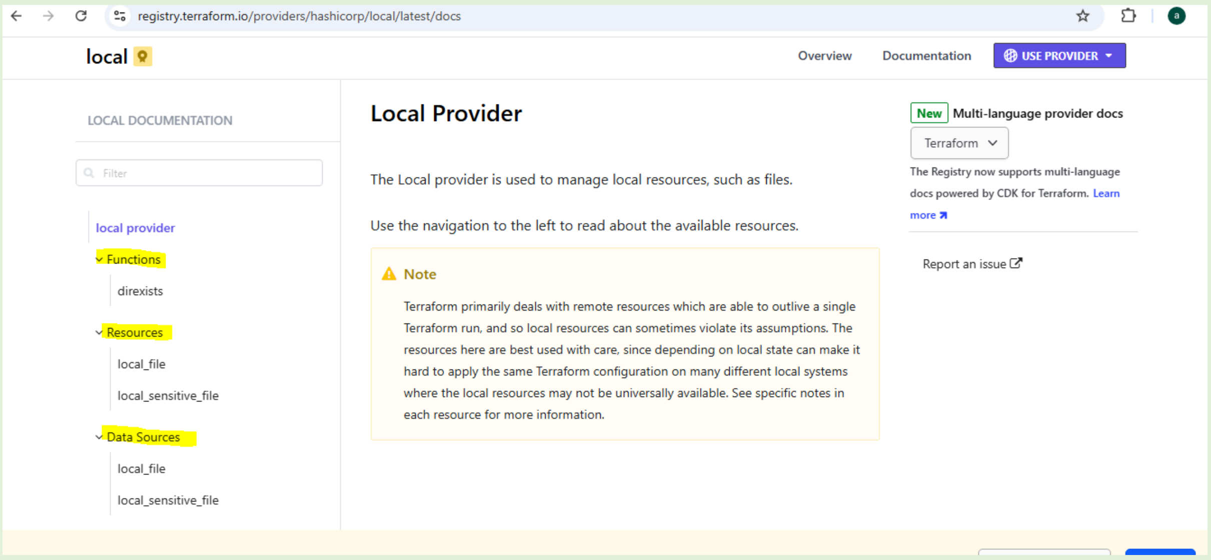
Use the navigation to the left to read about the available resources.

Note

Terraform primarily deals with remote resources which are able to outlive a single Terraform run, and so local resources can sometimes violate its assumptions. The resources here are best used with care, since depending on local state can make it hard to apply the same Terraform configuration on many different local systems where the local resources may not be universally available. See specific notes in each resource for more information.

The Registry now supports multi-language docs powered by CDK for Terraform. Learn more

We can check are the services Local Provider let you


registry.terraform.io/providers/hashicorp/local/latest/docs

local

Overview Documentation USE PROVIDER

LOCAL DOCUMENTATION

- local provider
- Functions
  - direxists
- Resources
  - local_file
  - local_sensitive_file
- Data Sources
  - local_file
  - local_sensitive_file

Local Provider

The Local provider is used to manage local resources, such as files.

Use the navigation to the left to read about the available resources.

Note

Terraform primarily deals with remote resources which are able to outlive a single Terraform run, and so local resources can sometimes violate its assumptions. The resources here are best used with care, since depending on local state can make it hard to apply the same Terraform configuration on many different local systems where the local resources may not be universally available. See specific notes in each resource for more information.

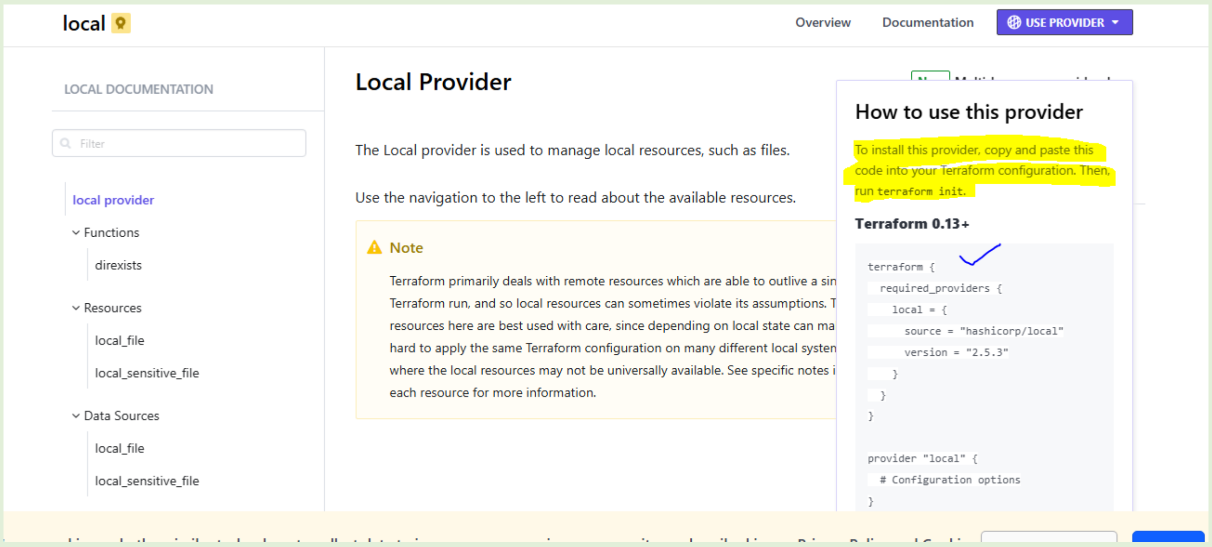
New Multi-language provider docs

Terraform

The Registry now supports multi-language docs powered by CDK for Terraform. Learn more

Report an issue

It will also let us know the configuration file with current version

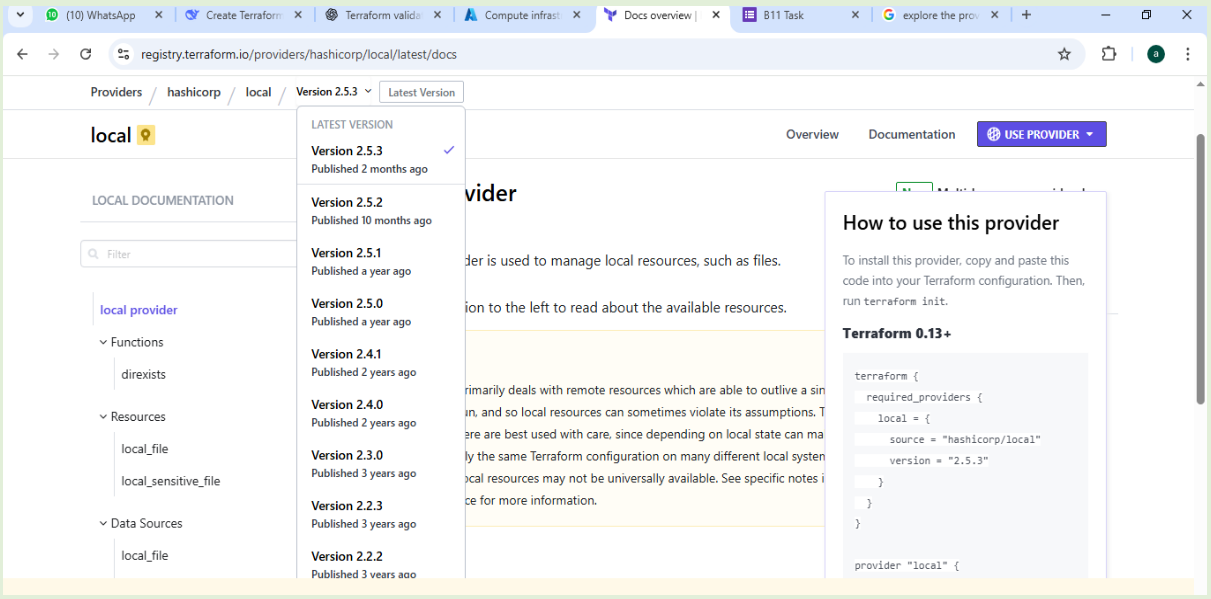

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    local = {
      source = "hashicorp/local"
      version = "2.5.3"
    }
  }
}

provider "local" {
  # Configuration options
}
```

Here we can use the versions which are provided if suppose I want the previous version I can have that version like we have certain

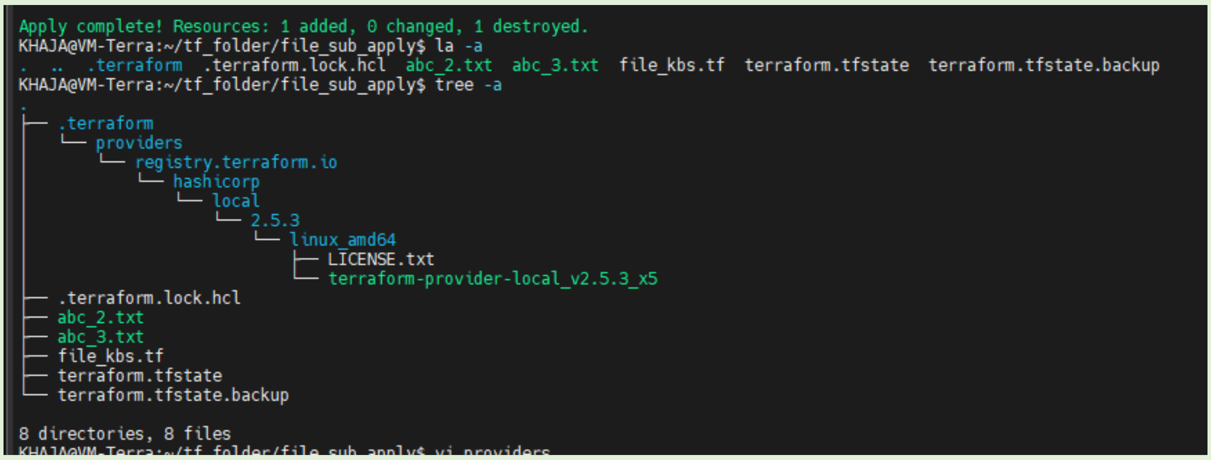

registry.terraform.io/providers/hashicorp/local/latest/docs

Providers / hashicorp / local / Version 2.5.3 Latest Version

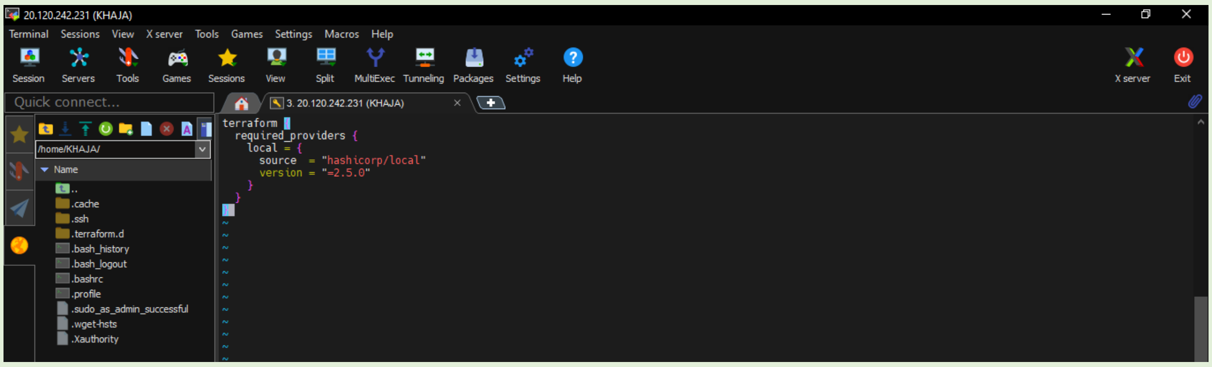
LATEST VERSION

Version 2.5.3 ✓	Published 2 months ago
Version 2.5.2	Published 10 months ago
Version 2.5.1	Published a year ago
Version 2.5.0	Published a year ago
Version 2.4.1	Published 2 years ago
Version 2.4.0	Published 2 years ago
Version 2.3.0	Published 3 years ago
Version 2.2.3	Published 3 years ago
Version 2.2.2	Published 3 years ago

I have the version like version 2.5.3 know I want to change to version 2.5.0



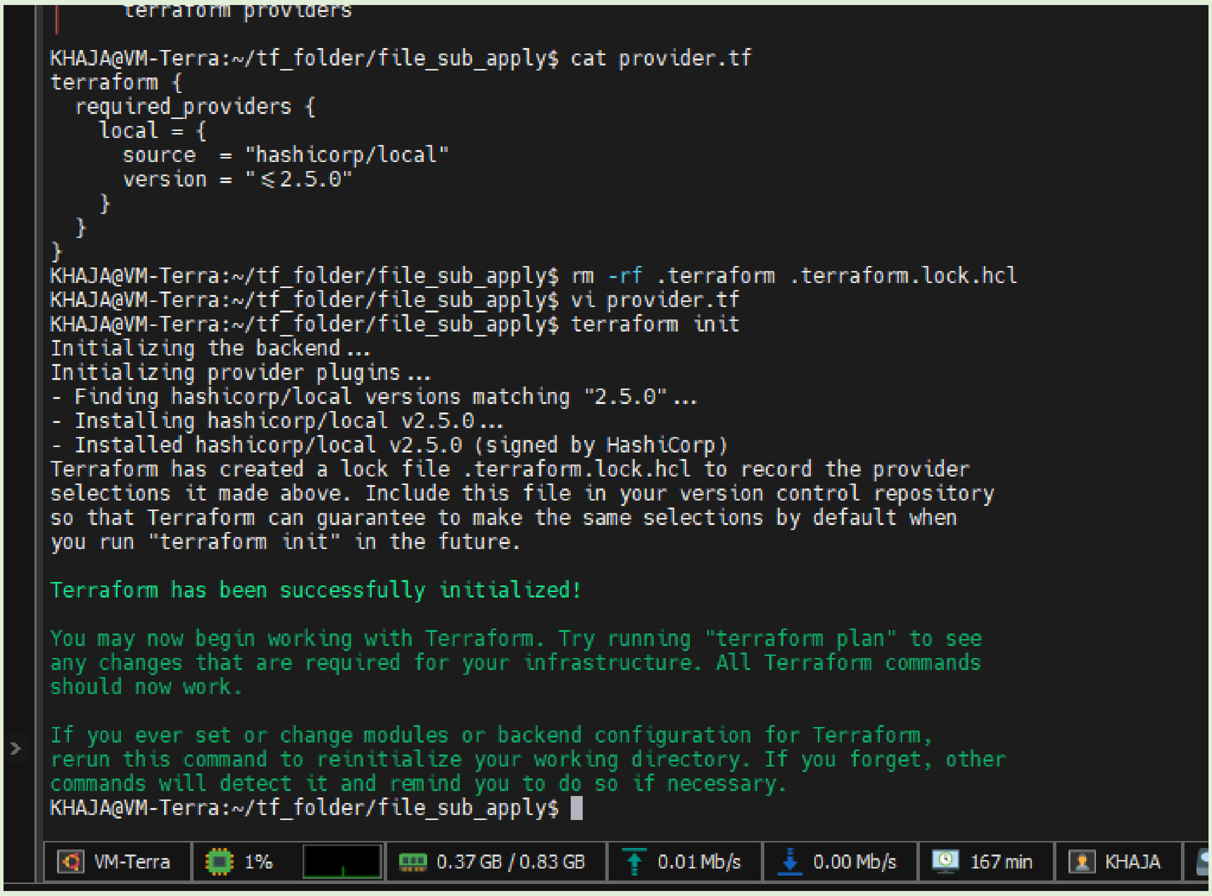
```
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ la -a
.  ..  .terraform  .terraform.lock.hcl  abc_2.txt  abc_3.txt  file_kbs.tf  terraform.tfstate  terraform.tfstate.backup
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ tree -a
.
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               └── local
│                   └── 2.5.3
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-local_v2.5.3_x5
├── .terraform.lock.hcl
├── abc_2.txt
├── abc_3.txt
├── file_kbs.tf
├── terraform.tfstate
└── terraform.tfstate.backup

8 directories, 8 files
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ vi providers
```


20.120.242.231 (KHAJA)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
3. 20.120.242.231 (KHAJA)
/home/KHAJA/
Name
..
.cache
.ssh
.terraform.d
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.wget-hsts
.Xauthority

```
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "=2.5.0"
    }
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
```

Run the command to Initialize Terraform

➢ terraform init



```
      terraform providers

KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ cat provider.tf
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "≤2.5.0"
    }
  }
}
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ rm -rf .terraform .terraform.lock.hcl
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ vi provider.tf
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/local versions matching "2.5.0"...
- Installing hashicorp/local v2.5.0...
- Installed hashicorp/local v2.5.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

Here the version has change form latest version 2.5.3 to previous version 2.5.0 as per the requirement

2. random **Provider**

Purpose: Generates random values (strings, numbers, pets) for unique resource naming or secrets.
Common Use Cases:

- Creating unique resource names (e.g., S3 buckets, VM hostnames).
- Generating temporary passwords/keys.

Key Resources

a) random_string

```
resource "random_string" "bucket_suffix" {
  length  = 8
  upper   = false
  special = false  # No special chars (e.g., `-`, `_`)
}
```

Output: bucket-name-3a7b9c2d

b) random_pet (Human-readable random names)

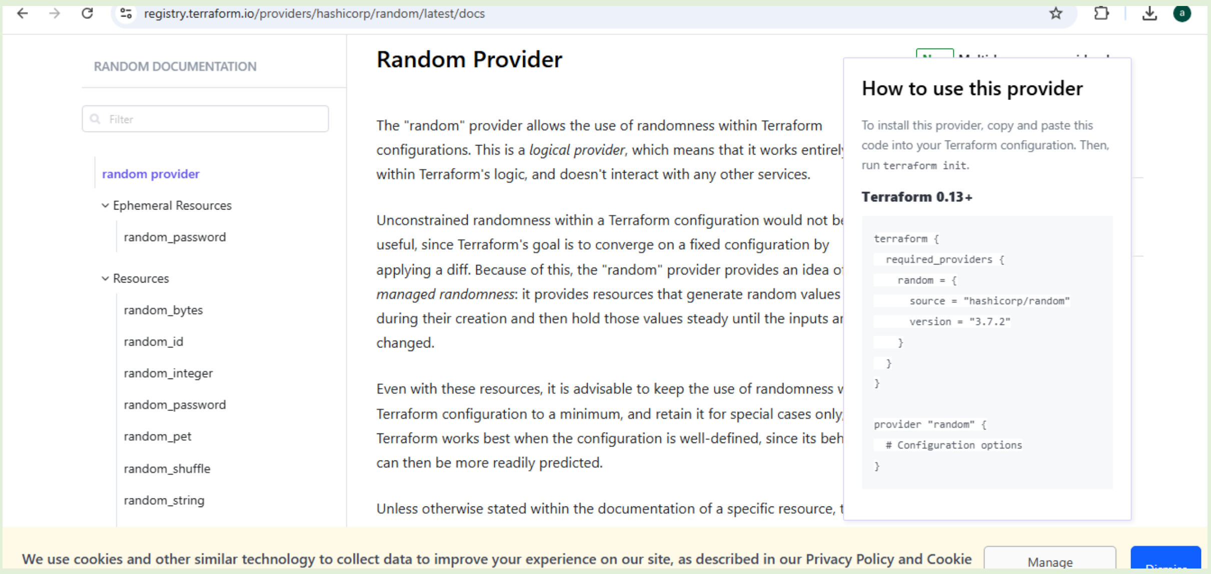
```
resource "random_pet" "server_name" {
  length = 2  # e.g., "happy-lemur"
}
```

Output: happy-lemur

c) random_password **(Sensitive)**

```
resource "random_password" "db_password" {
  length  = 16
  special = true
}
```

Note: Mark outputs as sensitive to hide them in logs.



registry.terraform.io/providers/hashicorp/random/latest/docs

RANDOM DOCUMENTATION

- random provider
- Ephemeral Resources
  - random_password
- Resources
  - random_bytes
  - random_id
  - random_integer
  - random_password
  - random_pet
  - random_shuffle
  - random_string

Random Provider

The "random" provider allows the use of randomness within Terraform configurations. This is a *logical provider*, which means that it works entirel within Terraform's logic, and doesn't interact with any other services.

Unconstrained randomness within a Terraform configuration would not be useful, since Terraform's goal is to converge on a fixed configuration by applying a diff. Because of this, the "random" provider provides an idea o *managed randomness*: it provides resources that generate random values during their creation and then hold those values steady until the inputs ar changed.

Even with these resources, it is advisable to keep the use of randomness v Terraform configuration to a minimum, and retain it for special cases only Terraform works best when the configuration is well-defined, since its beh can then be more readily predicted.

Unless otherwise stated within the documentation of a specific resource, t

How to use this provider

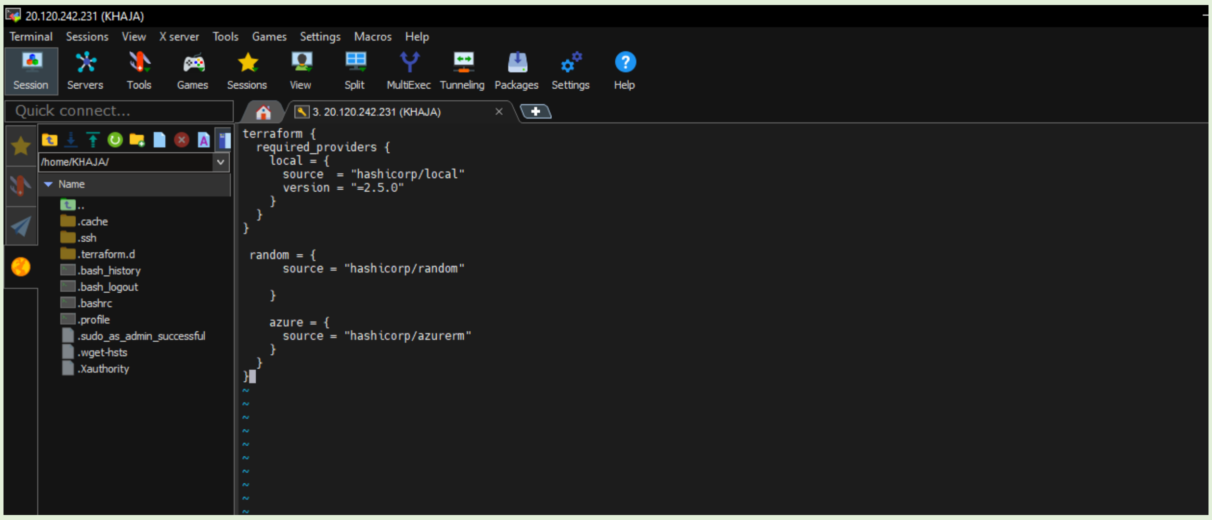
To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    random = {
      source = "hashicorp/random"
      version = "3.7.2"
    }
  }
}

provider "random" {
  # Configuration options
}
```

We use cookies and other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie

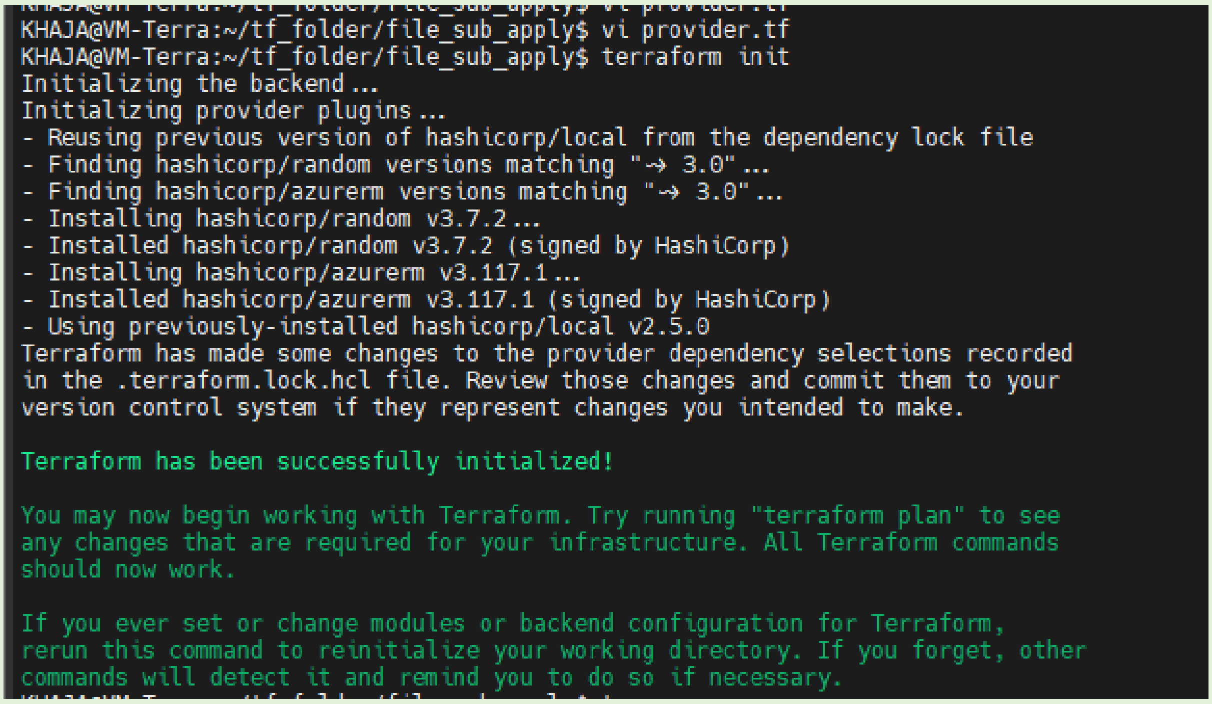
When we use the provider like


20.120.242.231 (KHAJA)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
3. 20.120.242.231 (KHAJA)
/home/KHAJA/
Name
..
.cache
.ssh
.terraform.d
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.wget-hsts
.Xauthority

```
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "=2.5.0"
    }
  }
}

 random = {
      source = "hashicorp/random"

    }

    azure = {
      source = "hashicorp/azurerm"
    }
  }
}
~
~
~
~
~
~
~
~
~
~
```

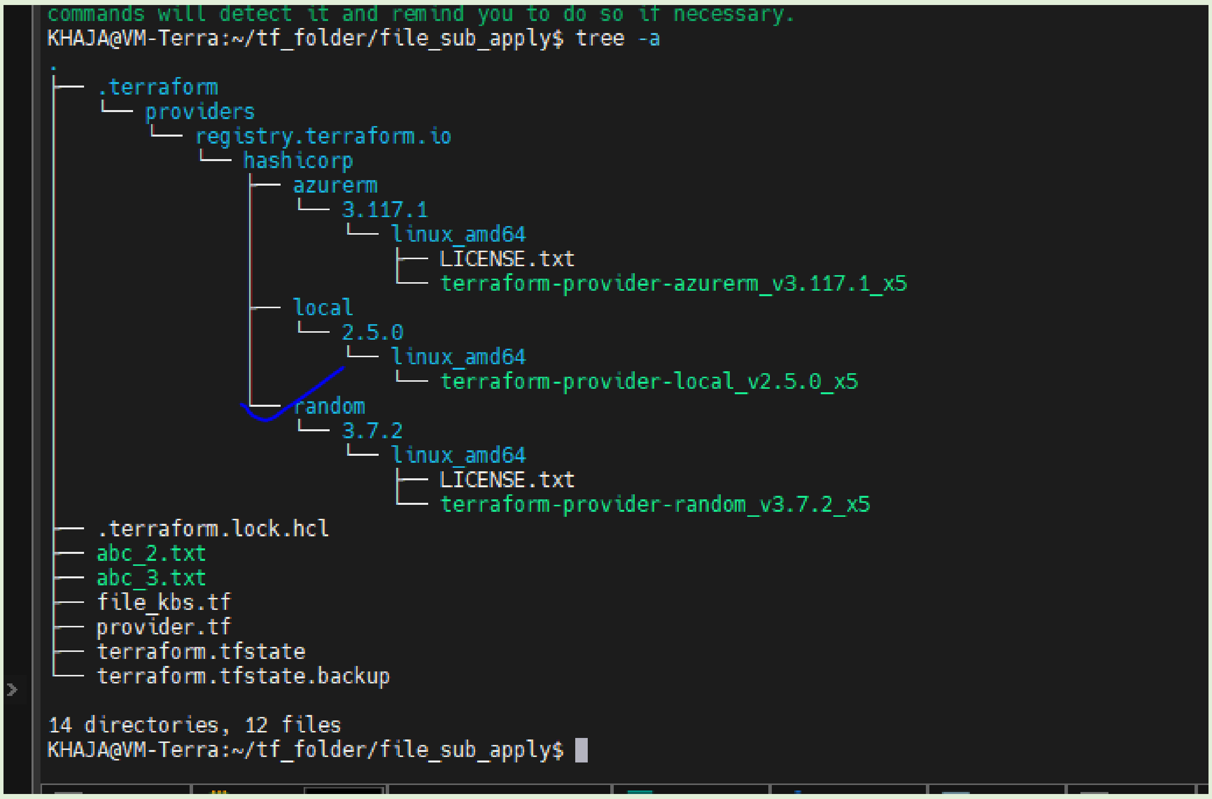
Execute the command init

Terraform init



```
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ vi provider.tf
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Finding hashicorp/random versions matching "~> 3.0"...
- Finding hashicorp/azurerm versions matching "~> 3.0"...
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)
- Installing hashicorp/azurerm v3.117.1...
- Installed hashicorp/azurerm v3.117.1 (signed by HashiCorp)
- Using previously-installed hashicorp/local v2.5.0
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Check the tree for version



```
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ tree -a
.
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               ├── azurerm
│               │   └── 3.117.1
│               │       └── linux_amd64
│               │           ├── LICENSE.txt
│               │           └── terraform-provider-azurerm_v3.117.1_x5
│               ├── local
│               │   └── 2.5.0
│               │       └── linux_amd64
│               │           └── terraform-provider-local_v2.5.0_x5
│               └── random
│                   └── 3.7.2
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-random_v3.7.2_x5
├── .terraform.lock.hcl
├── abc_2.txt
├── abc_3.txt
├── file_kbs.tf
├── provider.tf
├── terraform.tfstate
└── terraform.tfstate.backup

14 directories, 12 files
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```

3. Null Provider

Provides constructs that intentionally do nothing – useful in various situations to help orchestrate tricky behavior or work around limitations.

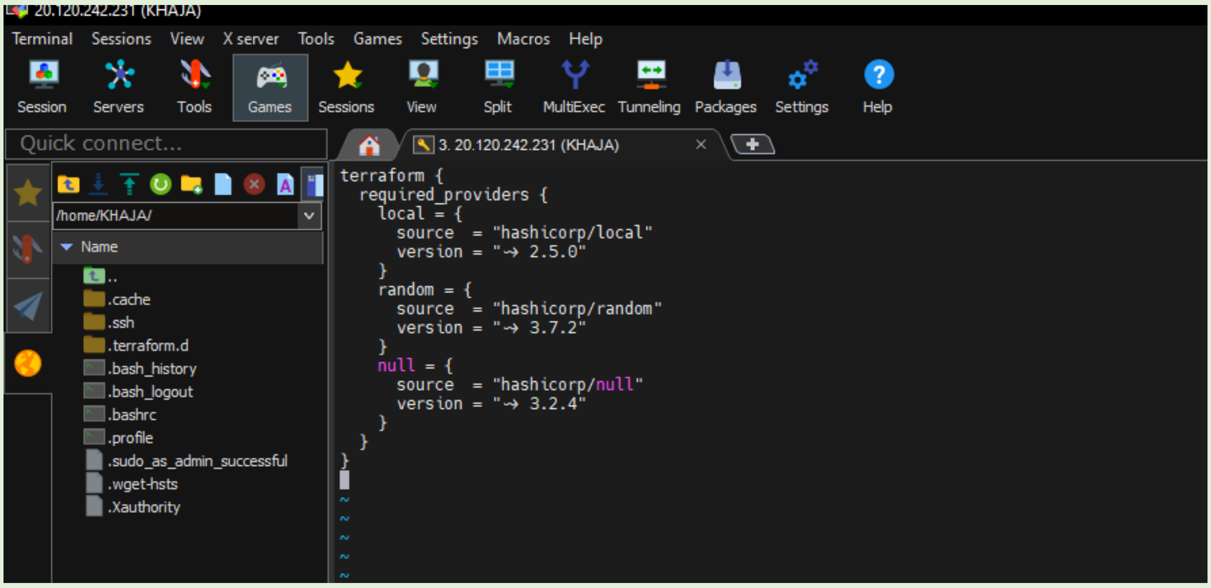
The null provider is a rather-unusual provider that has constructs that intentionally do nothing. This may sound strange, and indeed these constructs do not need to be used in most cases, but they can be useful in various situations to help orchestrate tricky behavior or work around limitations.

The documentation of each feature of this provider, accessible via the navigation, gives examples of situations where these constructs may prove useful.

Usage of the null provider can make a Terraform configuration harder to understand. While it can be useful in certain cases, it should be applied with care and other solutions preferred when available.

Make a configuration

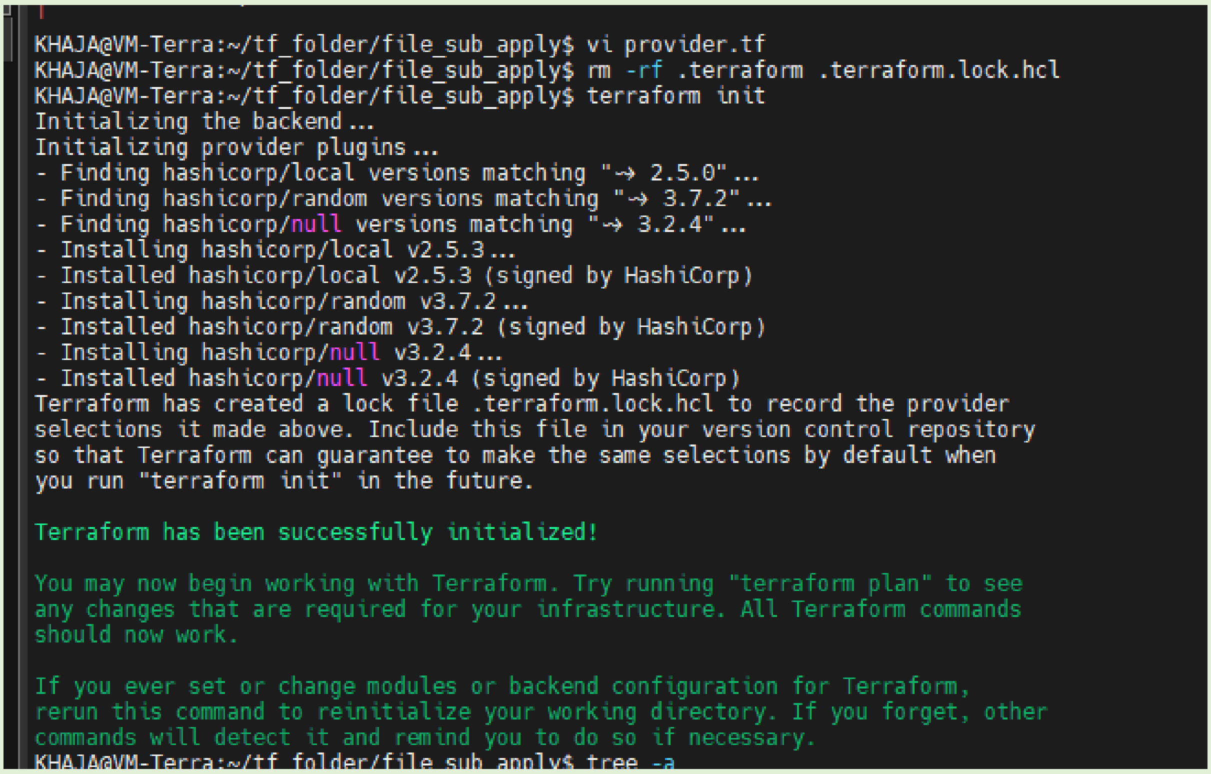
```
terraform {
  required_providers {
    null = {
      source = "hashicorp/null"
      version = "3.2.4"
    }
  }
}
```


20.120.242.231 (KHAJA)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
3. 20.120.242.231 (KHAJA)
/home/KHAJA/
Name
..
.cache
.ssh
.terraform.d
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.wget-hsts
.Xauthority

```
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "~> 2.5.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "~> 3.7.2"
    }
    null = {
      source  = "hashicorp/null"
      version = "~> 3.2.4"
    }
  }
}
~
~
~
~
~
```

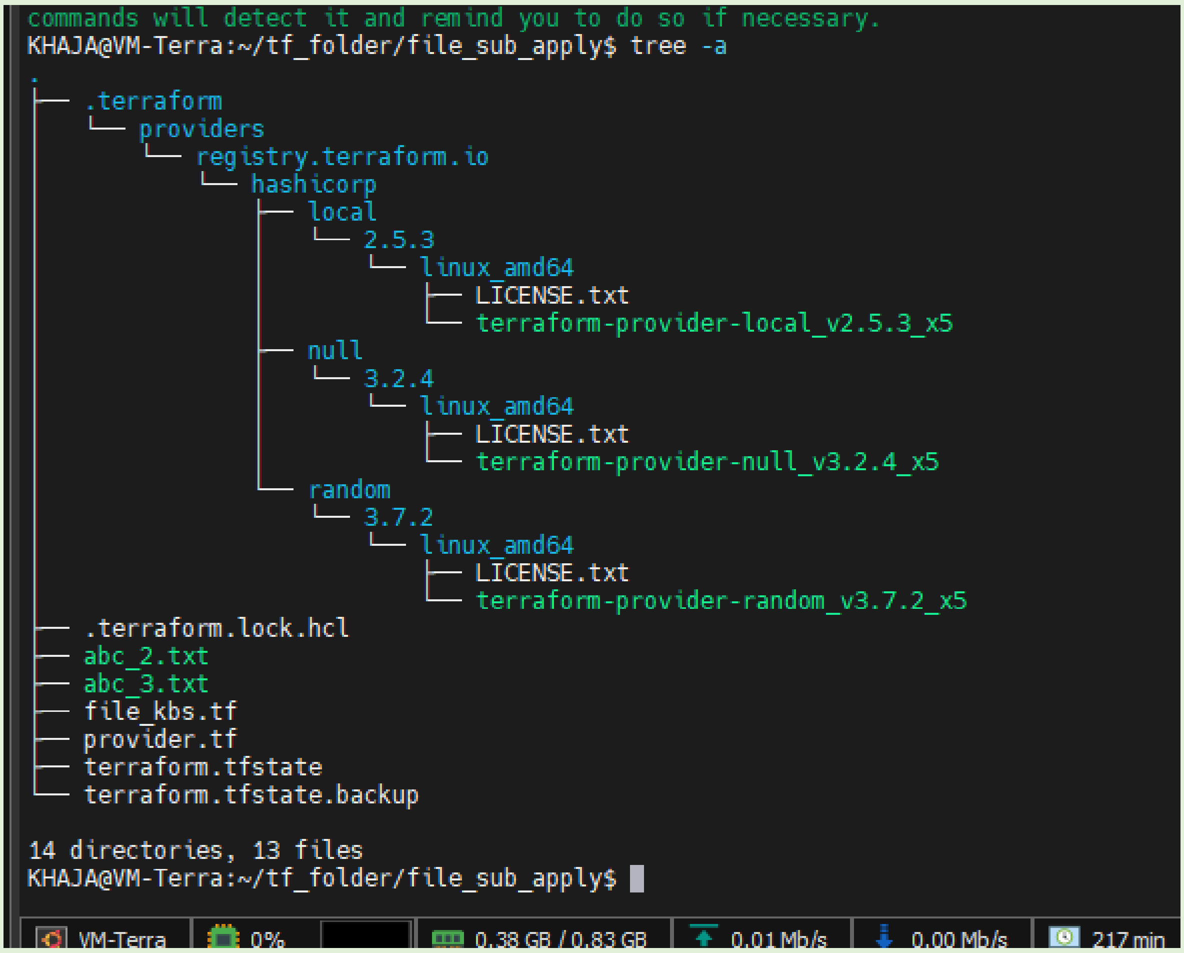
Execute the command init

Terraform init



```
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ vi provider.tf
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ rm -rf .terraform .terraform.lock.hcl
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/local versions matching "~> 2.5.0"...
- Finding hashicorp/random versions matching "~> 3.7.2"...
- Finding hashicorp/null versions matching "~> 3.2.4"...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)
- Installing hashicorp/null v3.2.4...
- Installed hashicorp/null v3.2.4 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ tree -a
```

Check the contents of directory using tree -a



```
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$ tree -a
.
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               ├── local
│               │   └── 2.5.3
│               │       └── linux_amd64
│               │           ├── LICENSE.txt
│               │           └── terraform-provider-local_v2.5.3_x5
│               ├── null
│               │   └── 3.2.4
│               │       └── linux_amd64
│               │           ├── LICENSE.txt
│               │           └── terraform-provider-null_v3.2.4_x5
│               └── random
│                   └── 3.7.2
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-random_v3.7.2_x5
├── .terraform.lock.hcl
├── abc_2.txt
├── abc_3.txt
├── file_kbs.tf
├── provider.tf
├── terraform.tfstate
└── terraform.tfstate.backup

14 directories, 13 files
KHAJA@VM-Terra:~/tf_folder/file_sub_apply$
```